APPARATUS AND METHOD FOR GENERATING PACKED SUM OF

ABSOLUTE DIFFERENCES

by

Daniel W. J. Johnson

Albert J. Loper Jr.

Assignee: VIA Technologies, Inc.
              8F, 533, Chung-Cheng Road
              Hsin-Tien, Taipei 231, Taiwan

Address correspondence to:

Huffman Law Group, P.C.
Customer Number 23,669
1832 N. Cascade Ave.
Colorado Springs, CO   80907
719.475.7103
719.623.0141
jim@huffmanlaw.net

APPARATUS AND METHOD FOR GENERATING PACKED SUM OF

ABSOLUTE DIFFERENCES

by

Daniel W. J. Johnson

Albert J. Loper Jr.

PRIORITY INFORMATION

[0001]   This application claims priority based on U.S. Provisional Application, Serial No. 60/444531, filed January 31, 2003, entitled APPARATUS AND METHOD FOR GENERATING PACKED SUM OF ABSOLUTE DIFFERENCES.

FIELD OF THE INVENTION

[0002]   This invention relates in general to the field of arithmetic operations in microprocessors and particularly to the generation of packed sums of absolute differences.

BACKGROUND OF THE INVENTION

[0003]   The x86 instruction set includes a PSADBW instruction. The PSADBW instruction includes two 64-bit input operands, each arranged as eight packed unsigned byte integers. One of the operands is a minuend operand of a subtraction operation and the other operand is a subtrahend operand of the subtraction operation. The PSADBW instruction generates an unsigned 16-bit result which is the sum of the absolute value of the eight differences of the corresponding eight unsigned byte integers when subtracting the subtrahend from the minuend. This particular result must be computed in various common

applications, such as multimedia audio, video, or graphics applications, or scientific applications.

[0004]   One   approach   to   implementing   the   PSADBW instruction in a microprocessor is to generate the differences of the first and second packed operands, then take the absolute value of the differences, and then serially add the absolute values of the differences. However, this approach has the drawback of requiring a relatively large number of processor clock cycles to generate the result, particularly because the adds are performed serially.   Therefore, what is needed is a fast apparatus for performing the PSADBW instruction.

SUMMARY

[0005]   In one aspect, the present invention provides an apparatus that performs an MMX PSADBW instruction in a microprocessor with an MMX unit.   The apparatus includes carry-generating packed subtraction logic that subtracts the packed unsigned subtrahend operand from the packed unsigned minuend operand to generate packed differences and a carry bit associated with each difference. A Boolean one carry bit indicates the difference is negative and a Boolean zero carry bit indicates the difference is positive. The apparatus includes a mux for each difference that selects the difference if the carry is a zero (i.e., is positive) or its complement if the carry is a one (i.e., is negative). The selectively inverted differences and the carry bits are added in parallel to produce the instruction result.   In the case of a negative difference, by adding the Boolean one carry bit to the inverse of the difference, the absolute value of the difference is obtained.   In the

case of a positive difference, by adding the Boolean zero carry bit to the non-inverted difference, the absolute value of the difference is obtained. Advantageously, the absolute value computation is accomplished substantially in parallel with the addition of the selectively inverted differences and carry bits to perform a fast PSADBW result.

[0006]   In one embodiment, the MMX unit includes two 16-bit multiplier pipelines. The multiplier pipelines each include a partial product generator that generates nine partial products by Booth encoding. Addition logic of the apparatus includes a carry-save adder in each multiplier pipeline for adding the partial products to generate a sum. The addition logic also includes a full adder for adding the two sums from the two multiplies. The apparatus includes a mux in each multiplier pipeline that selects multiplier partial products for provision to the addition logic in the case of a multiply instruction. However, in the case of a PSADBW instruction, the muxes select the selectively inverted differences and the carry bits for provision to the addition logic.

[0007]   In one embodiment, the microprocessor translates the PSADBW macroinstruction into first and second microinstructions for execution by the microprocessor's MMX unit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008]   FIGURE 1 is a block diagram illustrating a prior art block diagram of the MMX PSADBW instruction.

[0009]   FIGURE 2 is a block diagram illustrating a microprocessor for executing a PSADBW instruction according to the present invention.

[0010]   FIGURE 3 is a block diagram illustrating the MMX unit of Figure 2 according to the present invention.

[0011]   FIGURE 4 is a flowchart illustrating operation of the microprocessor of Figure 2 to perform a PSADBW instruction according to the present invention.

DETAILED DESCRIPTION

[0012]   Referring now to Figure 1, a block diagram illustrating a prior art block diagram of the MMX PSADBW instruction 100 is shown. The MMX PSADBW instruction 100 includes an instruction opcode 102 uniquely specifying the PSADBW instruction 100, and two instruction operands 104 and 106. The first operand 104 comprises a minuend operand of eight packed unsigned bytes denoted X0 through X7. The second operand 106 comprises a subtrahend operand of eight packed unsigned bytes denoted Y0 through Y7. The instruction 100 generates a result 108 which is the sum of the absolute value of the eight differences of the corresponding eight unsigned byte integers of the minuend operand 104 and the subtrahend operand 106. The PSADBW instruction is described in detail in the 1999 Intel® Architecture Software Developer's Manual, Volume 2: Instruction Set Reference, at pages 3-545 through 3-547, which are hereby incorporated by reference.

[0013]   Referring now to Figure 2, a block diagram illustrating a microprocessor 200 for executing a PSADBW instruction according to the present invention is shown. The microprocessor 200 includes instruction translation logic 202, a microinstruction queue 204 coupled to the instruction translation logic 202, and an MMX unit 206 coupled to the microinstruction queue 204.

[0014]   The instruction translation logic 202 translates a macroinstruction, such as the PSADBW macroinstruction 100 of Figure 1, into one or more microinstructions. In one embodiment, the macroinstructions comprise instructions of the x86 instruction set, such as MMX instructions. In one embodiment, the instruction translation logic 202 translates a PSADBW instruction 100 into two microinstructions, denoted PMULSAD 212 and PSUBSAD 214 in Figure 2. The PSUBSAD microinstruction 214 instructs the MMX unit 206 to generate the differences of the packed PSADBW operands, to generate the carry bit of each of the differences, and to selectively invert each of the differences based on the value of its associated carry bit. The PMULSAD microinstruction 212 instructs the MMX unit 206 to add the carry bits and selectively inverted differences to generate the PSADBW instruction result. The operation of the PSUBSAD 214 and PMULSAD 212 microinstructions is described in more detail below with respect to Figures 3 and 4.

[0015]   The instruction translation logic 202 comprises logic,   circuits,   devices,   or   microcode   (i.e., microinstructions or native instructions), or a combination of logic, circuits, devices, or microcode, or equivalent elements that are employed to translate instructions into associated sequences of microinstructions. The elements employed to perform translation within the instruction translation logic 202 may be shared with other circuits, microcode,   etc.,   that   are   employed   to   perform   other functions   within   the   microprocessor   200.   A microinstruction (also referred to as a native instruction)

is an instruction at the level that an execution unit executes, such as MMX unit 206. For example, microinstructions are directly executed by a reduced instruction set computer (RISC) microprocessor. For a complex instruction set computer (CISC) microprocessor such as an x86-compatible microprocessor, x86 instructions are translated into associated microinstructions, and the associated microinstructions are directly executed by a unit or units within the CISC microprocessor.

[0016] The microinstructions are provided by the instruction translation logic 202 to the microinstruction queue 204 for storage pending execution by execution units of the microprocessor 200, such as MMX unit 206. The microinstruction queue 204 has a plurality of microinstruction entries. Microinstructions are provided from the microinstruction queue 204 to execution units of the microprocessor 200, such as MMX unit 206.

[0017] In one embodiment, the MMX unit 206 includes an MMX register file having a plurality of registers for storing instruction operands, such as the PSADBW instruction minuend operand 104 and subtrahend operand 106 of Figure 1. The MMX unit 206 executes the operations prescribed by microinstructions as passed down from previous stages of microprocessor 200. The MMX unit 206 comprises logic, circuits, devices, or microcode (i.e., microinstructions or native instructions), or a combination of logic, circuits, devices, or microcode, or equivalent elements that are employed to perform operations as prescribed by microinstructions provided thereto. The elements employed to perform the operations within the MMX

unit 206 may be shared with other circuits, microcode, etc., that are employed to perform other functions within the microprocessor 200. In one embodiment, the MMX unit 206 operates in parallel to other execution units such as an integer unit, floating point unit, etc. In one embodiment that is compatible with the x86 architecture, the MMX unit 206 operates in parallel with an x86 integer unit, an x86 floating point unit, and an x86 SSE® unit. According to the scope of the present application, an embodiment is compatible with the x86 architecture if the embodiment can correctly execute a majority of the application programs that are designed to be executed on an x86 microprocessor. An application program is correctly executed if its expected results are obtained. Alternative x86-compatible embodiments contemplate the MMX unit 206 operating in parallel with a subset of the aforementioned x86 execution units. The MMX unit 206 is described in more detail with respect to Figures 3 and 4 below.

[0018] Referring now to Figure 3, a block diagram illustrating the MMX unit 206 of Figure 2 according to the present invention is shown. The MMX unit 206 includes carry-generating packed subtraction logic 308 that receives a microinstruction 306, such as a PMULSAD 212 or PSUBSAD 214 microinstruction, from the microinstruction queue 204 of Figure 2. The subtraction logic 308 also receives the PSADBW instruction minuend operand 104 and the subtrahend operand 106 of Figure 1. The subtraction logic 308 includes subtractors that generate packed unsigned byte differences 314 for each of the corresponding packed unsigned byte minuend/subtrahend pairs. The differences

314 are denoted X7-Y7 through X0-Y0 in Figure 3. The differences 314 are generated using two's complement arithmetic.

[0019] The subtraction logic 308 also generates carry bits 312 for each of the differences 314. The carry bits 312 are referred to as C7 through C0 in Figure 3. In one embodiment, the carry bits 312 are stored in storage elements, such as latches or registers. The carry bits 312 are also referred to as borrow bits, or underflow bits, or sign bits, since they indicate whether a borrow from what would be the ninth bit of the minuend 104 was required and indicate whether the associated difference 314 is positive or negative. For each of the differences 314, its associated carry bit 312 is a Boolean value one (1) if the difference 314 is negative and is a Boolean value zero (0) if the difference 314 is positive. For example, assume the minuend X4 104 is 13_{10}, or 00001101_2 and the subtrahend Y4 106 is 9_{10}, or 00001001_2. Then the difference X4-Y4 314 is 4_{10}, or 00000100_2, and the carry bit C4 312 is 0 because no borrow was necessary. That is, the carry bit C4 312 indicates that the difference X4-Y4 314 is positive, and the difference X4-Y4 314 is also the absolute value of the difference X4-Y4 314. However, assume the minuend X4 104 is 9_{10}, or 00001001_2 and the subtrahend Y4 106 is 13_{10}, or 00001101_2. Then the difference X4-Y4 314 is -4_{10}, or 11111100_2, and the carry bit C4 312 is 1 because a borrow was necessary. That is, the carry bit C4 312 indicates that the difference X4-Y4 314 is negative, and the difference X4-Y4 314 is not the absolute value of the difference X4-Y4 314. In order to obtain the absolute

value of the difference X4-Y4 314, according to two's complement arithmetic, the difference 314 must be inverted and have 1 added to it. In the example, inverting the difference 314 and adding 1 would yield 00000100_2, or 4_{10}, which is the absolute value of the difference -4_{10}, or 11111100_2.

[0020] The MMX unit 206 also includes eight byte inverters 316, each coupled to receive a corresponding one of the packed differences 314 and generates a Boolean complement, or inverse, thereof.

[0021] The MMX unit 206 also includes eight two-input multiplexers 318, each coupled to a corresponding one of the byte inverters 316. Each multiplexer 318 receives the output of its corresponding inverter 316 on one input and receives the corresponding difference 314 on its other input. Each multiplexer 318 receives as its control input a corresponding carry bit 312. If the carry bit 312 is a Boolean zero, then the multiplexer 318 selects the difference 314 input; if the carry bit 312 is a Boolean one, then the multiplexer 318 selects the inverted difference 314 input from the inverter 316. Thus the output of the multiplexers 318 comprise eight selectively inverted differences 322 of the differences 314 generated by the subtraction logic 308. The selectively inverted differences 322 are denoted Z7 through Z0 322 in Figure 3.

[0022] The MMX unit 206 also includes two 16-bit multiplier pipelines, referred to as multiplier A 338A and multiplier B 338B in Figure 3. Each of the multipliers 338 includes a partial product generator, referred to in Figure 3 as multiplier A partial product generator 336A and

multiplier B partial product generator 336B, which generate multiplier A partial products 334A and multiplier B partial products 334B, respectively.   In one embodiment, the partial product generators 336 comprise Booth encoders, each of which generates nine 16-bit partial products based on examination of 3 bits of the multiplier operand to generate each partial product.   In one embodiment, the partial products also include additional bits, such as sign-extension bits. Hence, advantageously, at least eight of the partial products have at least eight bits of overlap during their addition.

[0023]   The multipliers 338A and 338B also include two-input multiplexers, referred to as addend mux A 326A and addend mux B 326B, respectively, in Figure 3. Addend mux A 326A receives on one input the partial products 334A from partial product generator 336A. Addend mux A 326A receives on its other input the carry bits 312. Addend mux B 326B receives on one input the partial products 334B from partial product generator 336B. Addend mux B 326B receives on its other input the selectively inverted differences 322. Each of the addend muxes 326 receives on its control input a microinstruction type signal 324, which indicates whether the microinstruction type is a PMULSAD microinstruction or a multiply instruction.   If the microinstruction type 324 indicates a PMULSAD microinstruction is being executed, then addend mux 326A selects the carry bits 312 to output; otherwise, addend mux 326A selects the partial products 334A to output. If the microinstruction type 324 indicates a PMULSAD microinstruction is being executed, then addend mux 326B

selects the selectively inverted differences 322 to output; otherwise, addend mux 326B selects the partial products 334B to output.

[0024]   The multipliers 338A and 338B also include adders 328A and 328B, respectively. In one embodiment, the adders 328 comprise carry-save adders. Adder 328A receives the output of addend mux A 326A and adder 328B receives the output of addend mux B 326B. That is, adder 328A receives either the partial products 334A as addends for adding together or adder 328A receives the carry bits 312 for adding together, based on the value of the microinstruction type signal 324. Similarly, adder 328B receives either the partial products 334B as addends for adding together or adder 328B receives the selectively inverted differences 322 for adding together, based on the value of the microinstruction type signal 324. In one embodiment, advantageously, the adders 328 are configured to add at least nine addends, each comprising at least 16-bits. In particular, all of the addends overlap at least eight bits, and the sum generated by the adders 328 comprises 32 bits. In one embodiment, each of the adders 328 comprises a first row of 3:2 carry-save adders (CSAs) that reduce the nine partial products to six intermediate partial products, a second row of 3:2 CSAs that reduce the six partial products to four intermediate partial products, a third row of CSAs that reduces the four partial products to three partial products, and a fourth row of CSAs that reduce the three partial products to two intermediate partial products, comprising a carry and a sum.

[0025]   MMX unit 206 also includes an adder 332 that receives the sums generated by adders 328A and 328B.   In one embodiment, adder 332 comprises a full adder.   Adder 332 adds the sums received from adders 328A and 328B and adds them to generate a final sum, which is the result 108 of the PSADBW instruction if the microinstruction type signal 324 indicated a PMULSAD microinstruction, or otherwise the sum of products of two 16-bit multiplies.   In one embodiment, if the desired result is a 16-bit multiply product result, then a zero value is muxed into one of the inputs of adder 332 (i.e., the input that is not from the multiplier performing the 16-bit multiply) to generate a final 16-bit multiply product.   In one embodiment, the two multipliers 338 operate in conjunction along with adder 332 to perform a 32-bit multiply.

[0026]   In one embodiment, the MMX unit 206 also includes two carry-save adders (not shown) that reduce the sum and carry from adder 328A and sum and carry from adder 328B to a single sum and carry that are provided to full adder 332, which generates the single final sum 108.

[0027]   As may be seen from the foregoing description, advantageously the generation of the absolute value of each of the packed differences 314 is accomplished concurrently with the addition of the differences 314 by virtue of the concurrent addition of the carry bits 312 and the selectively inverted differences 322.   That is, if a given difference 314 is negative, the absolute value is accomplished by selecting the inverse of the difference 314 by the multiplexer 318, and then effectively adding the carry bit 312, (which is a Boolean one if the difference

314 is negative) to the inverted difference, since the carry bit 312 will have a Boolean one value if the inverted difference is selected as the selectively inverted difference 322. Conversely, if a given difference 314 is positive, the absolute value is accomplished by selecting the non-inverted difference 314 by the multiplexer 318, and then effectively adding the carry bit 312, (which is a Boolean zero if the difference 314 is positive) to the non-inverted difference 314, since the carry bit 312 will have a Boolean zero value if the non-inverted difference 314 is selected as the selectively inverted difference 322. Advantageously, the addition of the carry bits 312 and the selectively inverted differences 322 is performed in parallel, rather than generating the absolute values of the differences and then serially adding the absolute values, thereby producing a fast PSADBW instruction result.

[0028]   Referring now to Figure 4, a flowchart illustrating operation of the microprocessor 200 of Figure 2 to perform a PSADBW instruction according to the present invention is shown. Flow begins at block 402.

[0029]   At block 402, the microprocessor 200 fetches a PSADBW macroinstruction 100 of Figure 1. Flow proceeds to block 404.

[0030]   At block 404, the instruction translation logic 202 of Figure 2 translates the PSADBW instruction into PSUBSAD 214 and PMULSAD 212 microinstructions, which are stored into the microinstruction queue 204 and forwarded to the MMX unit 206 of Figure 2. Flow proceeds to block 406.

[0031]   At block 406, the MMX unit 206 executes the PSUBSAD microinstruction 214. In particular, in response

to the PSUBSAD microinstruction 214, the subtractors in the subtraction logic 308 of Figure 3 subtract the subtrahend 106 from the minuend 104 to generate eight packed differences 314 and their associated carry bits 312 of Figure 3. A Boolean one carry bit 312 value indicates a negative difference, whereas a Boolean zero carry bit 312 value indicates a positive difference. Flow proceeds to block 408.

[0032] At block 408, the byte inverters 316 invert the differences 314 and the muxes 318 select the inverted difference if the corresponding carry bit 312 is a Boolean one and select the difference 314 if the carry bit 312 is a Boolean zero to generate the selectively inverted differences 322 of Figure 3. Flow proceeds to block 412.

[0033] At block 412, the microinstruction type 324 is provided to the addend muxes 326 of Figure 3. Flow proceeds to decision block 414.

[0034] At block 414, the addend muxes 326 determine whether the microinstruction type 324 is a PMULSAD microinstruction 212. If so, flow proceeds to block 422; otherwise, flow proceeds to block 416.

[0035] At block 416, the addend muxes 326 select the partial products 334 from the partial product generators 336 of Figure 3. Flow proceeds to block 418.

[0036] At block 418, the adders 328 and 332 of Figure 3 add the partial products 334 to produce the multiply instruction result. Flow ends at block 418.

[0037] At block 422, the addend muxes 326 select the eight selectively inverted differences 322 and the eight carry bits 312. Flow proceeds to block 424.

[0038]   At block 424, the adders 328 and 332 add the eight selectively inverted differences 322 and the eight carry bits 312 to produce the PSADBW 100 instruction result 108. Flow ends at block 424.

[0039]   As may be seen from the foregoing description, the present invention advantageously performs the PSADBW instruction in two microinstruction cycles.   In one embodiment, the PSADBW macroinstruction 100 is performed in four core clock cycles of the microprocessor 200. That is, the MMX unit 206 performs the packed subtraction and selective inversion of the differences according to the PSUBSAD microinstruction 214 in a single core clock cycle and performs the addition of the carry bits 312 and selectively inverted differences 322 according to the PMULSAD microinstruction 212 in three core clock cycles.

[0040]   As may be seen from the foregoing description, the multiplexers 326 enable the MMX unit 206 to advantageously employ the adders 328 and 332 to selectively perform the PSADBW and multiply instructions, thereby potentially reducing the total amount of circuitry needed to perform multiple instructions through reuse of common circuitry.

[0041]   Although the present invention and its objects, features and advantages have been described in detail, other embodiments are encompassed by the invention. For example, although embodiments have been described with respect to a 64-bit packed operand version of the MMX PSADBW instruction, embodiments are contemplated in which the apparatus is extended to a version of the PSADBW that operates on 128-bit packed operands. Furthermore, although

the invention has been described with respect to Boolean zero and Boolean one values, the Boolean values may be represented in various ways within the circuit elements, in particular using different logic level and voltage or current conventions, without departing from the scope of the invention.   Finally, the microprocessor may include multiple MMX execution units.

[0042]   Also, in addition to implementations of the invention using hardware, the invention can be implemented in computer readable code (e.g., computer readable program code, data, etc.) embodied in a computer usable (e.g., readable) medium.   The computer code causes the enablement of the functions or fabrication or both of the invention disclosed herein.   For example, this can be accomplished through the use of general programming languages (e.g., C, C++, JAVA, and the like); GDSII databases; hardware description languages (HDL) including Verilog HDL, VHDL, Altera HDL (AHDL), and so on; or other programming and/or circuit (i.e., schematic) capture tools available in the art.   The computer code can be disposed in any known computer usable (e.g., readable) medium including semiconductor memory, magnetic disk, optical disk (e.g., CD-ROM, DVD-ROM, and the like), and as a computer data signal embodied in a computer usable (e.g., readable) transmission medium (e.g., carrier wave or any other medium including digital, optical or analog-based medium).   As such, the computer code can be transmitted over communication networks, including Internets and intranets. It is understood that the invention can be embodied in computer code (e.g., as part of an IP (intellectual

property) core, such as a microprocessor core, or as a system-level design, such as a System on Chip (SOC)) and transformed to hardware as part of the production of integrated circuits. Also, the invention may be embodied as a combination of hardware and computer code.

[0043]   Finally, those skilled in the art should appreciate that they can readily use the disclosed conception and specific embodiments as a basis for designing or modifying other structures for carrying out the same purposes of the present invention without departing from the spirit and scope of the invention as defined by the appended claims.

We claim: